THE BELL SYSTEM
TECHNICAL JOURNAL

VOLUME XXXVIII JANUARY 1959 NUMBER 1

Logic for a Digital Servo System

By R. W. KETCHLEDGE

(Manuscript received June 27, 1958

Methods are described for performing comparisons of fast binary numbers. These techniques have proved useful in the positioning of cathode ray tube beams in a photographic memory. A binary address is compared with a digital indication of the present position in circuitry called digital servo logic. The output of the servo logic is an analog indication of the positional error. Logics are described for obtaining sign only, sign plus magnitude and sign plus approximate magnitude.

1. INTRODUCTION

Digital storage of information on photographic emulsion is characterized by the large amounts of information that can be stored on a small physical area.[1] This same advantage also implies the need for exceptionally high precision in the access facilities. A memory system of this type has been developed which uses a cathode ray tube to interrogate simultaneously a group of photographic plates.[2,3,4] The access problem in this store is to position an electron beam in accordance with a binary number to an accuracy of a fraction of a thousandth of an inch with microsecond positioning times. The binary address calls for digital circuitry, while the high accuracy implies the use of feedback techniques.

The positioning technique adopted uses a fraction of the storage capacity to indicate, in parallel digital form, the present position of the

cathode ray spot. This digital indication is then compared with the binary number representing the desired position. The comparison is performed in circuitry called the digital servo logic. The output of the servo logic is an analog error signal which drives the electron beam to eliminate the positional error. The use of this feedback technique permits the beam position to be determined by the mechanical edges of bar patterns on a group of photographic plates. This relaxes the mechanical tolerances on the optical system and reduces the need for high precision in the electrical circuits.

The digital servo logic problem is a number comparison problem. The number representing the present position must be compared with the binary number representing the desired address in order to extract the sign of the difference or, preferably, the sign and magnitude of the difference. The number comparison must remain valid at all possible transition values of the present position number, since this number varies continuously during the servo process.

The transition problem is one of the more difficult aspects of digital servoing. The actual position takes on a continuous range of values during the servo operation and, consequently, one desires a continuous indication of the position, even though the position is being represented in "digital" form. Thus, transition values for the position digits must operate the circuit satisfactorily. In binary codes where many digits change simultaneously difficulties occur because these changes are not exactly simultaneous. This has led to the use of standard Gray code to represent the present position. In addition, a "pseudo-binary" translation of the Gray number is required.

Conventional digital adding and subtracting circuits are unsuitable for high positioning speeds because of their dependence upon digits of low significance. In normal subtraction logic the least significant digits are compared first and the carry process moves towards digits of greater significance. One departure of this digital servo logic from a conventional parallel subtractor is the use of carries that proceed from the most significant digit towards the least. This permits the subtraction process to ignore digits which are changing too rapidly to be read.

Finally, the output of the logic need not be digital. The typical requirement is only for an analog signal representing the number difference. In most cases, this signal is an error signal, and a rather rough approximation of its magnitude is sufficient. However, the analog signal must be a continuous representation of the error and, near the desired address, fractional cell errors must be corrected.

II. SIGN-ONLY LOGIC

2.1 *General*

Sign-only methods are less complex than the sign-plus-magnitude methods. Also, the sign-only methods are a useful introduction to some of the concepts involved. Thus they are described first. The simplest of these is binary-binary sign-only logic — logic that gives the sign of the difference between two numbers when both are expressed in conventional binary code.

2.2 *Binary-Binary Sign-Only*

The logic circuit described here takes two binary numbers as voltages or no voltage on two sets of leads and produces an output whenever the first number is equal to or larger than the second. Modifications can also be used to recognize as separate signals "larger," "equal" or "smaller."

The method is to observe the most significant error and to disregard all errors of lesser significance. This can be accomplished by comparing the numbers digit by digit, starting with the most significant digits and disregarding all but the first error found. In other words, the sign of the most significant mismatch indicates the sign of the difference between the two numbers.

The logic circuit of Fig. 1 is one which will perform the desired operations and yield an output if and only if $a_1a_2a_3 \cdots a_n$ is equal to or larger

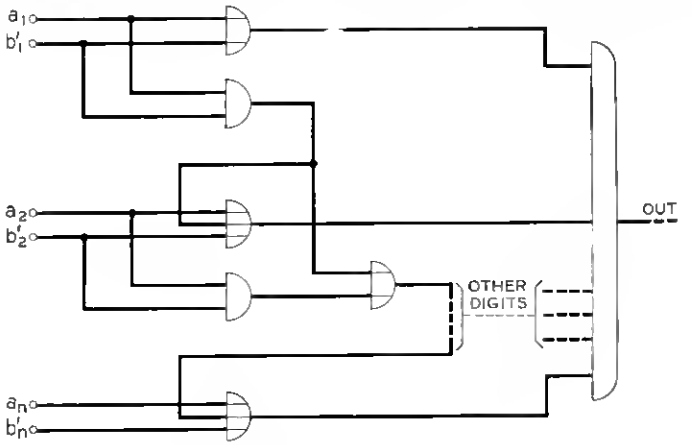

Fig. 1 — Comparison circuit for binary numbers.

than $b_1b_2b_3 \cdots b_n$. Note that this logic is driven by the logical complement of the b number. Thus, if the a digits are the output of the optical beam position decoder and the b digits are the address of the desired beam position, the deflection amplifier may be properly driven by the output of the logic circuit. This will servo the beam to the edge of the match position. The transition problem limits the practicality of this method for servo purposes where transitional values of one of the numbers must be faithfully decoded. This suggests the use of Gray code for the present-position numbers.

2.3 *Gray-Binary Sign-Only*

Having binary-binary logic indicates that, for Gray-binary, all that is needed is to translate the Gray to binary and then compare. However, the translation introduces the same transition difficulties that required the use of Gray code initially. Fortunately, the difficulty is avoidable by the use of a pseudo-translation of the Gray number.

To translate a Gray number to binary, reverse any Gray digit which is preceded by a 1 in the binary translation. Thus, a Gray 111 translates to a binary 101. The most significant Gray 1 is not reversed since it is "preceded" by a binary 0. Thus the most significant binary digit is a 1. This reverses the second Gray digit, making it 0 in binary. The least significant Gray digit is thus preceded by a 0 and is not reversed.

The normal translation of Gray to binary is based on a binary number that changes as the Gray number changes. The pseudo-translation of Gray to binary is based on a fixed binary number. It is useful because of the following, somewhat surprising, fact: Choose any Gray number and any binary number. Reverse those Gray digits which are immediately preceded by a I in the binary number. This forms a pseudo-binary number. If the original Gray number was larger than the binary number, the pseudo-binary number, interpreted as binary, will also be larger. If the Gray was smaller, the pseudo-binary will be smaller; if equal, equal. In other words, pseudo-translation of a Gray number to pseudo-binary does not change the sign of the comparison with the controlling binary. In the servo logic the binary address is used to reverse those Gray digits which immediately follow 1's in the address. The new number so formed can then be compared with the address in binary-binary logic to obtain the sign of the difference. Note that the pseudo-binary number is still characterized by only one digit transition at a time. Therefore, the transition difficulties associated with multiple simultaneous digit changes do not occur.

TABLE I

Present Position (Gray)	Desired Address (Binary)							
	000	001	010	011	100	101	110	111
000	000	000	001	001	010	010	011	011
001	001	001	000	000	011	011	010	010
011	011	011	010	010	001	001	000	000
010	010	010	011	011	000	000	001	001
110	110	110	111	111	100	100	101	101
111	111	111	110	110	101	101	100	100
101	101	101	100	100	111	111	110	110
100	100	100	101	101	110	110	111	111

Table I shows the pseudo-binary translation for three-digit numbers An investigation of Table I shows the following features:

i. Corresponding to a given input address, each column yields a code where only one digit changes at a time.

ii. For the numbers lying along the main diagonal in the table, the Gray number corresponding to the input address has been transformed into the input address number. Thus, a match between the pseudo-binary number and the input address indicates that the beam is at the desired address.

iii. All positions above the desired one are translated into numbers which, in a binary sense, are smaller than the input number; those below are translated into larger ones.

Property iii indicates that the final step merely involves determining whether a particular binary number is greater or smaller than a given one. A simple circuit which effects such a determination has already been presented. Fig. 2 shows a logic structure for such a Gray-binary comparison.

2.4 *Gray-Gray Sign-Only*

Logics have been found which compare two Gray numbers directly with polarity reversals controlled by the address. The sign of the difference is determined by the most significant mismatch between the numbers, except that the sign of this mismatch is reversed if there is an odd number of preceding 1's in the address. Alternatively, one can merely translate the address Gray number to binary, and then use Gray-binary logic.

A third method is to treat the Gray address as if it were a binary number. With random addressing, it often does not matter where the

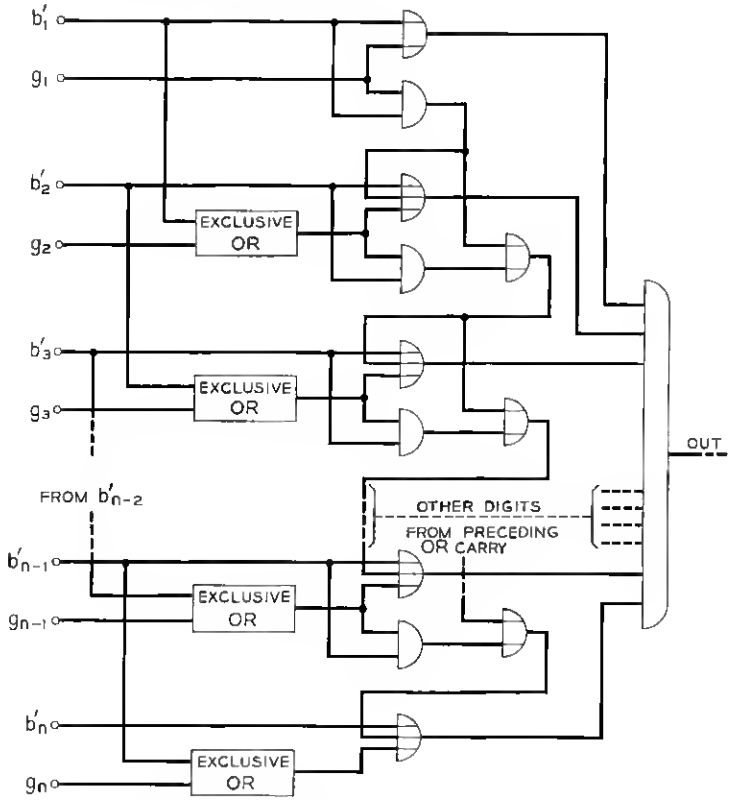

Fig. 2 — Gray-binary sign-only logic.

information is stored on the storage surface. Thus, transpositions occur in information location, but an independent position is still obtained for every address.

III. LOGIC FOR MAGNITUDE AND SIGN

3.1 *General Binary-Binary Considerations*

If both the address number and the position number are in binary code, a simple subtraction will establish the difference. In this subtraction,

$$1 - 1 = \quad 0,$$
$$1 - 0 = +1,$$
$$0 - 1 = -1,$$
$$0 - 0 = \quad 0.$$

This forms a difference number and three possible values for each of its digits, namely, $+1$, 0 or -1. If these digits are weighted as in ordinary binary notation (1, 2, 4, 8, etc.) and the indicated sign is applied, the resultant number is an accurate measure of the magnitude and sign of the difference. However, certain combinations lead to a plus digit followed by a succession of minus digits. This cancellation effect makes it very difficult to obtain an accurate analog voltage for the difference number. This accuracy problem makes it desirable, therefore, to consider schemes in which large cancellations are avoided.

In any of the magnitude-determining circuits, regardless of the codes used, some form of quantizing or offset should be added to define an exact balance point. An added digit can be used to permit a small additional drive that will settle the servo in the center of the zero-output region of the logic. Alternatively, a small fixed drive in a fixed direction can be used so that the net drive in the balance region of the logic is small but finite. Typically, it should be equal to a half-cell error. This would cause the servo to drive to the transition between a zero output for the logic and a one-cell output of sign opposite to the fixed drive. This latter is probably the easiest method of quantizing, since it does not require an additional digit and its corresponding circuitry.

3.2 *A Typical Binary-Binary Logic*

A number of binary-binary logics have been found for obtaining the analog magnitude and sign. In general, these compare the two numbers digit by digit, developing at each digit outputs and/or carries. The carries propagate towards digits of less significance. The outputs feed a digital-to-analog converter of a conventional sort. The direction of carry in certain cases can result in some outputs being developed from digits of high significance which represent too large an error signal. This is corrected by the following outputs being generated in the opposite sign. So long as this cancellation does not exceed 50 per cent, no large loss in accuracy results. In effect, this approach takes a $+$, 0, $-$ form of the binary difference number and translates it into a form where severe cancellations cannot occur. In the logic to be described next, all cancellations have been eliminated by increasing the possible outputs per digit from three ($+$, 0, $-$) to five (-2, -1, 0, $+1$, $+2$).

The point of departure is to take one binary number, subtract a second from it, and obtain a binary difference number whose digits may be $+1$, 0 or -1 but whose digits have the same magnitude significance as a conventional binary number. This binary difference number is then modified by logic circuits. Finally, conventional binary-to-analog

conversion is used, but each binary digit can drive separately in either plus or minus sign and in single or double strength. Therefore, the possible outputs on any binary position are $+2$, $+1$, 0, -1, -2. Plusses and minuses of output are never generated simultaneously on the comparison of any two binary numbers (no cancellations).

The first mismatch starts a carry, but this carry cannot be stopped by a succeeding mismatch of opposite sign. If the first mismatch (most significant) is plus, a plus carry is started. This inhibits the initiation of any minus carry in less significant digits. A following 0 (match) produces a single-strength plus output at that digit's weight, but a following minus mismatch inhibits this single-strength plus output. A following plus mismatch combined with the plus carry produces a second single-strength plus output and, since in this case the other output is not inhibited, a double-strength total is generated. Equivalent rules apply on a minus carry. The Boolean algebra representing these rules is given below. Each V_n represents a "unity" output, and double weight occurs when both V_n's are active; A = position digit, B = address digit:

$$
\begin{aligned}
(+\Delta B_n) &\equiv B_n A_n', \\
(-\Delta B_n) &\equiv B_n' A_n , \\
(\text{following } C_+) &= (+\Delta B)(C_+')(C_-') + C_+ = C_+ + (+\Delta B)(C_-'), \\
(\text{following } C_-) &= (-\Delta B)(C_+')(C_-') + C_- = C_- + (-\Delta B)(C_+'), \\
+V_{n_1} &= C_+(-\Delta B_n)', \\
-V_{n_1} &= C_-(+\Delta B_n)', \\
+V_{n_2} &= C_+(+\Delta B_n), \\
-V_{n_2} &= C_-(-\Delta B_n).
\end{aligned}
$$

A logic circuit having these characteristics is shown on Fig. 3.

3.3 *A Use for Binary-Binary Logic*

Transition problems limit the usefulness of binary-binary logics when transitional values must be faithfully decoded. However, it can be used as an applique in a sign-only servo. A sign-only servo is relatively slow for large address changes because of the limited error signal. If more speed is desired, it is possible to add a forward-acting positioning circuit. This can be a conventional digital-to-analog converter driven directly from the address, in which case the servo merely mops up for converter inaccuracies. Alternatively, binary-binary logic can be used to add to

the existing deflection voltage an additional voltage proportional to the change in address. In this case, the old binary address is compared with the new address, and approximate positioning occurs on a forward-acting basis. The use of binary-binary logic to obtain an analog indication of the address change has an accuracy advantage over the use of the new address alone for small address changes. This is because the digital-to-analog converter is required to convert a smaller number when decoding only the change in address.

3.4 *General Gray-Binary Considerations*

The comparison of a binary and a Gray number can take a variety of forms, but, as indicated earlier, the objective of high speed has sharply limited the techniques considered. The problem again is to find a fast, simple method, avoiding carries except toward digits of less significance,

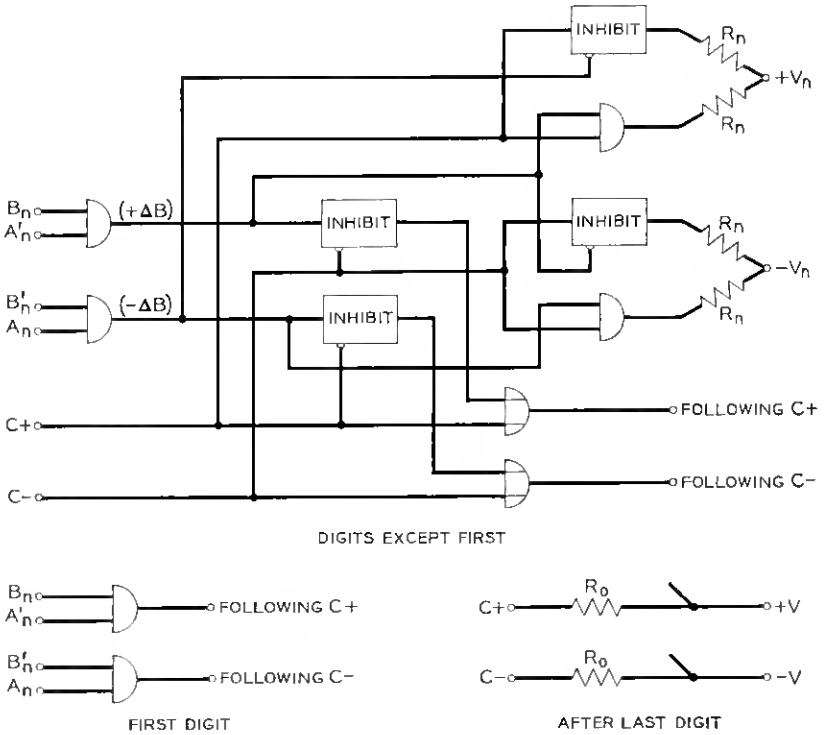

Fig. 3 — Binary-binary logic for magnitude and sign.

etc. Several methods have been devised. All of the schemes derive from the same general observation: merely an observation of the geometrical properties of a Gray code and the numerical significance of a change in a single digit.

Consider a conventional Gray code and what happens to its numerical value if the first digit is reversed. If the initial number were 100 (7), the inversion produces 000 (0) and a change in value of 7 in the example chosen. If the initial number were 110 (4), the inversion yields 010, or 3, and the change is only 1. Inspection of the geometry of the Gray code points up that the position in the sequence of numbers images about the natural reversal point of the digit in question. For this three-digit Gray code, the first digit changes between 3 and 4, and taking any number in the sequence and reversing the first digit merely moves the number to the image point and then an equal distance on the other side. Thus, changing from 100 or 7 to 000 or 0 means moving from 7 to $3\frac{1}{2}$ and then moving the same distance past the image point. Therefore, if we know how far we are from the image point to start with, we know that the digit change represents twice this distance. The obvious conclusion is that, for a change in a single digit of a Gray code, the change is twice the distance from the image point.

A further consideration of the geometric properties of the Gray code indicates its symmetry about the image point of the preceding digit. Take the image point of any digit and observe that the following Gray digits are symmetical about the image point. They can thus be used to measure the distance from the image point and consequently the effect of reversing the preceding digit. The distance to the image point is merely the following Gray digits interpreted as Gray but with the first following digit reversed. If one uses the binary translation of the Gray number being changed, it is possible to use it as a measure of the distance to the image point in the Gray code. Thus, the following binary digits (or their complement) are also a measure of the image distance. Therefore, the magnitude significance of a change in a Gray digit can be obtained from either the following Gray or following binary digits.

None of the foregoing has considered the important case where more than one digit is changed at once. However, this is a matter of signs rather than magnitudes and is described best in connection with the particular schemes. The important point is that, if one is given two Gray numbers which differ in only a single digit, the magnitude of the difference is obtainable from either the following Gray digits or from the corresponding binary digits of one of the numbers.

3.5 *A Typical Gray-Binary Servo Logic*

Determination of the mismatches can be obtained by a comparison of the two numbers in Gray code. It can also be done in the following way. Use the binary address to pseudo-translate the Gray position to pseudo-binary. Then compare this pseudo-binary number with the binary address. The mismatches will occur in the same digits, although the directions of the mismatches (1/0 vs. 0/1) will be reversed following a 1 in the binary address. Recall that this same pseudo-translation was used in the binary-Gray method for sign alone. This produces a set of mismatches having values of +, −, 0, as shown in Table II.

The difference between the address and position is developed by adding the components of the difference represented by each mismatch. This requires the determination of a sign and a magnitude for each mismatch. Table III lists the rules for the magnitudes.

The weights of the mismatches are obtained as noted above following the properties described in Section 3.3. The signs of the mismatches are reversed following an odd number of preceding mismatches. This is not as bad as it might seem. Consider the first mismatch (most significant). A plus mismatch has a weight equal to the following binary address digits in double their binary weight plus one. In other words, gate out

TABLE II

Address	Binary / Gray	111 / 100	110 / 101	101 / 111	100 / 110	011 / 010	010 / 011	001 / 001	000 / 000
Position (Gray)	100	0 0 0	0 0 −	0 − +	0 − 0	− + 0	− + −	− 0 +	− 0 0
	101	0 0 +	0 0 0	0 − 0	0 − −	− + +	− + 0	− 0 0	− 0 −
	111	0 + +	0 + 0	0 0 0	0 0 −	− 0 +	− 0 0	− − 0	− − −
	110	0 + 0	0 + −	0 0 +	0 0 0	− 0 0	− 0 −	− − +	− − 0
	010	+ + 0	+ + −	+ 0 +	+ 0 0	0 0 0	0 0 −	0 − +	0 − 0
	011	+ + +	+ + 0	+ 0 0	+ 0 −	0 0 +	0 0 0	0 − 0	0 − −
	001	+ 0 +	+ 0 0	+ − 0	+ − −	0 + +	0 + 0	0 0 0	0 0 −
	000	+ 0 0	+ 0 −	+ − +	+ − 0	0 + 0	0 + −	0 0 +	0 0 0

TABLE III

Gray Position (after reversal if preceding $B_n = 1$)	Binary Address	Mismatch	Weight
1	1	0	0
0	0	0	0
0	1	+	1 + twice following binary number
1	0	−	1 + twice the complement of the following binary number

TABLE IV

First Mismatch	Second Mismatch	Reverse	Action
+	−	yes	double next output, stop carry
+	+	yes	stop carry

all following binary 1's into a converter with binary weighting and add 1. Now consider the second mismatch. If the first and second mismatches are of the same sign they subtract, because the sign of the second is reversed. This says, "do not energize any more outputs beyond the second mismatch." If the signs oppose, the reversal of the second says, "put out in the same polarity both the following binary and its complement." But that is the same as putting out the next following binary weight at double value or the corresponding one at normal. The latter is used here. Table IV may clarify this point.

The corresponding case for an initial minus is obvious.

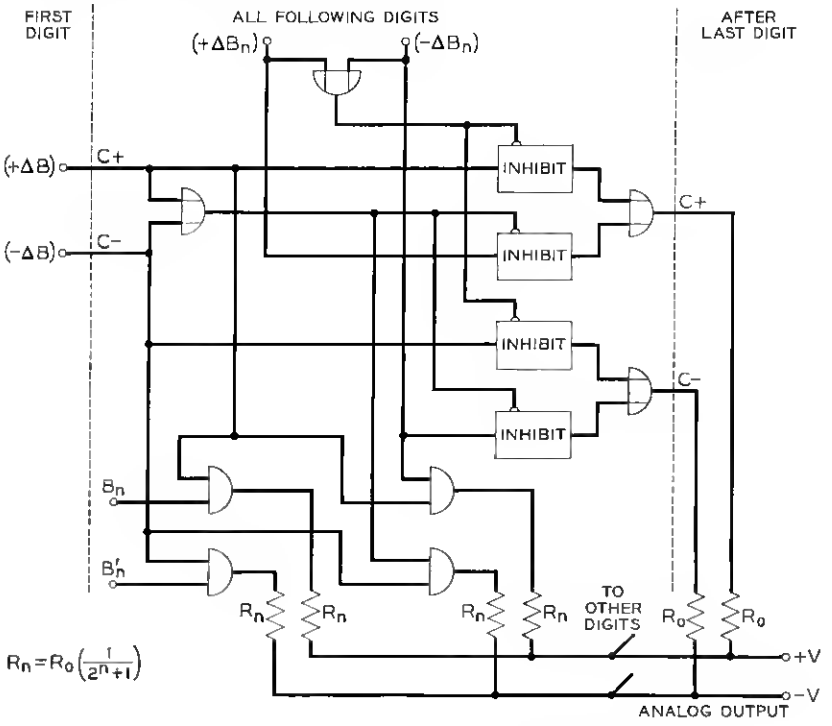

Fig. 4 — Gray-binary logic for magnitude and sign.

This leads rather simply to the following logic and the circuit of Fig. 4, in which $(+\Delta B)$ and $(-\Delta B)$ are plus and minus mismatches respectively:

$$\begin{aligned}
(\text{following } C_+) &= C_+(+\Delta B_n)'(-\Delta B_n)' + (C_-')(C_+')(+\Delta B_n),\\
(\text{following } C_-) &= C_-(+\Delta B_n)'(-\Delta B_n)' + (C_+')(C_-')(-\Delta B_n),\\
(+\Delta B_n) &= G_n(B_{n+1})\, B_n + G_n'(B_{n+1}')\, B_n,\\
(-\Delta B_n) &= G_n'(B_{n+1})\, B_n' + G_n(B_{n+1}')B_n',\\
(+V_{n1}) &= C_+B_n,\\
(-V_{n1}) &= C_-B_n',\\
(+V_{n2}) &= C_+(-\Delta B_n),\\
(-V_{n2}) &= C_-(+\Delta B_n).
\end{aligned}$$

Alternatively,

$$\begin{aligned}
(+V_n) &= C_+[B_n + (-\Delta B_n)],\\
(-V_n) &= C_-[B_n' + (+\Delta B_n)].
\end{aligned}$$

The first digit has no preceding carries so

$$\begin{aligned}
(\text{following } C_+) &= (+\Delta B),\\
(\text{following } C_-) &= (-\Delta B),\\
(+V_{n1}) &= (-V_{n1}) = (+V_{n2}) = (-V_{n2}) = 0.
\end{aligned}$$

The last digit (least significant) must be followed by a ± 1 contributor but no digit mismatch can occur, so

$$\begin{aligned}
(+V_0) &= C_+,\\
(-V_0) &= C_-.
\end{aligned}$$

3.6 *Other Gray-Binary and Gray-Gray Logics*

As noted in Section 3.4, the magnitude significance of Gray-to-Gray or pseudo-binary-to-binary mismatches can be determined from the following Gray digits. These digits can be used in a variety of ways to derive appropriate analog signals. Other logic structure variations involve the number of carries used and their significance. Additional variations depend upon the form of the drives to the digital to analog converter — three-valued, five-valued, with or without cancellation,

etc. It is also possible to form a set of analog signals and switch these to a common output under control of the mismatches. This particular approach probably is of value only in relay circuitry, because of the need for switching a wide range of analog signals.

In spite of all these possible variations there are common factors. Some of these are: the use of mismatches located by Gray-to-Gray or pseudo-binary-to-binary, carries which propagate toward digits of less significance, digital outputs which are controlled by mismatches of digits of equal or greater significance, use of the following digits of one of the numbers to develop the magnitude significance of a mismatch, creation of an analog signal accurately representing the number difference and accurate number comparisons even for transitional values of at least one of the numbers.

3.7 *The Edge Problem*

In any digital servo, and particularly in those using magnitude as well as sign, there is an edge problem. If the digital position information does not extend beyond the edge of the servo area, the circuit may fail if it gets out of the area. For example, a transient overshoot beyond the limits of the normal servo area should produce just as big an error signal as would a similar overshoot in the center of the servo area. Otherwise, overshoots beyond an edge may fail to produce any drive (or perhaps a very small drive) to return the beam to its desired edge position. The foregoing implies that it is mandatory to extend the coding of the position of the beam to the extreme limits to which the beam may be deflected. It is always possible to introduce limiters into the deflection circuit which will prevent extreme values of excess deflection. However, because of drift, these limiters must operate at some distance from the actual edge of the servo area.

One solution to the edge problem appears to be the following: Add one digit to the present position number in the most significant position, thereby extending it over double the normal range. This does not change the width of any digit but merely adds a digit in front of the previous number. The binary address is then modified as follows: The first (most significant) digit is moved one place in the more significant direction. This digit thereby is matched against the digit that was added to the code plate. The gap left in the binary address is filled by using the complement of the first binary address digit. It will be seen that this restricts the binary address so formed to numbers beginning with either 10 or 01. Thus, this binary sequence covers the center half of the range generated by the present position. With this arrangement, overshoots of up to 50

per cent in either direction will still generate accurate error signals. One other advantage is that the addition of a fixed half-cell drive provides full use of all address combinations.

IV. APPROXIMATE LOGICS

4.1 *General Considerations of Approximation*

All the magnitude-determining circuits mentioned in the preceding section are exact. This is a valuable property. However, an ordinary servo need not always determine the exact magnitude of the error. Often, only the approximate error is required. This may necessitate somewhat more loop gain margin, but this is not serious if the error variations are not too large. Much of the complexity of the exact error methods is attributable to the handling of signs. The signs of each of the individual contributions must be properly manipulated to obtain the correct total. Considerable simplification is possible if one merely determines the magnitude of the most significant error contributor and, in addition, the over-all sign of the error signal. In such cases, it is possible thereby to determine the over-all sign exactly and to determine the magnitude to within ± 6 db.

To approximate the difference between a Gray and a binary number to within 2-to-1 accuracy, it is sufficient to know the position and direction of the two most significant pseudo-binary-to-binary mismatches. If these first two mismatches are both of the same sign, only the most significant is necessary. When the first two mismatches are of opposite sign, the most significant mismatch dominates the sign, but the second most significant mismatch may sometimes dominate the magnitude. These considerations permit the construction of approximate magnitude and sign logics. Such logics can be derived independently or can be obtained by simplification of exact magnitude logics.

4.2 *A Typical Approximation Logic*

Fig. 5 shows one type of approximation logic. Here the sign and magnitude have been generated separately. A set of outputs, V_n, is energized to indicate the magnitude, although only the most significant V_n is to be used. This can be accomplished by decoding each V_n separately into an analog signal of the appropriate strength and feeding all of these analog signals through an ordinary diode OR circuit. The output of the OR circuit will equal the largest input and indicates the approximate magnitude of the error. Note that the digit weights used

are not quite binary. Instead of the usual 1, 2, 4, 8, etc., the weights follow the "1 + twice following binary" trend, namely, 1, 3, 5, 9, etc.

Since pseudo-translation of Gray does not change the mismatch pattern but only changes signs, a Gray-Gray method can be used with inputs composed of the binary address and the pseudo-binary position. This simplifies the determination of sign, since the sign is simply that of the most significant mismatch. The logic is:

$$\Delta_n = \text{mismatch of either sign},$$

$$C_{n-1} = C_n + \Delta_{n+1},$$

$$V_n = C_n G_n + \Delta_{n+1} G_n',$$

$$S_+ = \Delta_n B_n C_{n-1}' \text{ with OR from all digits.}$$

Note that G_n is the Gray position without any reversals by the binary address. The reversal of G_n by a preceding binary 1 is used only to form Δ.

Note also that only one carry is required and that only OR logic is used. This permits the carry circuit to be made of a group of cascaded cathode followers. This form of carry has been found to be extremely fast.

V. CONCLUSIONS

The type of digital servo logic described has permitted the attainment of precise high-speed positioning of electron beams. These logics use relatively few active elements compared with many other types of access to a comparable complex of addresses. This is because the logic

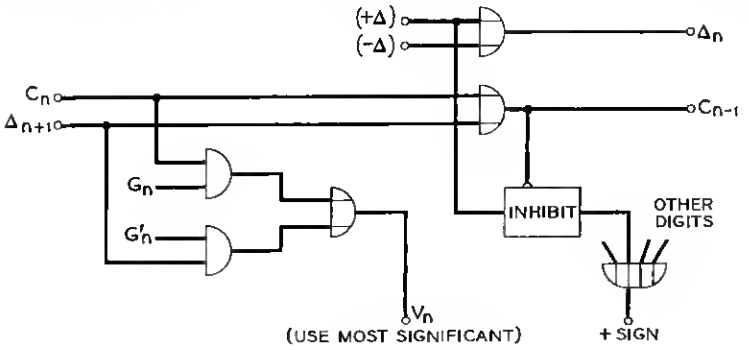

Fig. 5 — Approximation logic.

deals with the binary or Gray forms of the address instead of forms such as 1 out of N, which are less efficient.

These logics are characterized by carries that proceed toward digits of less significance. Thus, the digital outputs are not in the form of ordinary binary numbers but, in general, contain digits having more than two possible states. Such numbers are still suitable, however, for driving an ordinary digital-to-analog converter. Also, the weightings may depart from the normal binary values. A further characteristic of these logics is their ability to decode transitional values of Gray numbers. In other words, if the Gray digits are all 1 or 0 except one, and that digit is, say, 0.5, the logic still gives appropriate analog outputs. The key to this characteristic is the use of pseudo-binary translation of the Gray number. Finally, in the magnitude-determining logics, the following digits of the address are used to establish the magnitude significance of a mismatch.

VI. ACKNOWLEDGMENT

The author is indebted to his associates for both the experimental confirmation of these digital servo concepts as well as for numerous mathematical proofs and extensions. These concepts and various of their embodiments were obtained on a highly intuitive basis, and the author's associates were responsible for bringing mathematical coherence to a group of seemingly different logics. In particular the author wishes to mention L. E. Gallaher, M. Nesenbergs and V. O. Mowery.

REFERENCES

1. King, G. W., Brown, G. W. and Ridenour, L. N., Photographic Techniques for Information Storage, Proc. I.R.E., **41**, October 1953, p. 1421.
2. Staehler, R. E. and Davis, R. C., U. S. Patent No. 2,830,285.
3. Ketchledge, R. W., An Introduction to the Bell System's First Electronic Switching Office, Proc. of Eastern Joint Computer Conf., December 1957, p. 204.
4. Hoover, C. W., Jr., Staehler, R. E. and Ketchledge, R. W., Fundamental Concepts in the Design of the Flying Spot Store, B.S.T.J., **37**, September 1958, p. 1161.